Amendments to the Specification:

Please replace the paragraph beginning at page 2, line 4 with the following amended paragraph:

This ~~invention~~ disclosure relates to computer read pre-fetch operations~~,~~.

Please replace the paragraph beginning at page 2, line 8 with the following amended paragraph:

[.] Existing PCI bridges assist in the control of the sequencing of operations and access to computer busses in accordance with the bus specification (such as, for example, PCI Local Bus Specification Rev. 2.2 published by the PCI Special Interest Group). Pre-fetch algorithms are not covered by the PCI specification, but are widely employed by PCI devices to circumvent a fundamental issue with PCI protocol: it does not include a read amount embedded within each transaction. Such PCI devices employ a static read pre-fetch which requests the same amount of information for a particular type of read operation, regardless of the actual demands of the requesting agent. While this constant pre-fetch amount may be adjustable by means of a device specific configuration register, the selected amount is constant and applicable to all requesting agents served in connection with that register. A static pre-fetch amount may result in pre-fetching too much data.

Please replace the paragraph beginning at page 3, line 2 with the following amended paragraph:

Referring to FIG 1, an example adaptive ~~adaptive~~ read pre-fetch system 10 is shown having components on a bridge 12. The components include a pre-fetch factor register 15, being a re-writeable storage location. The adaptive read pre-fetch system 10 also includes a re-read pre-fetch factor register 20, a re-read timer 25 and a next read address register 28~~30~~. Also shown is pre-fetchable data storage such as system memory 40, and[ ] agents 35a, 35b and 35c. Each of the components of the adaptive read pre-fetch system 10 are preferably part of or attached to the computer, such as a bridge 12, within which the pre-fetch factor register 15, the re-read pre-fetch factor register 20, the re-read timer 25, and the next read address register 28~~30~~ may, but need not, reside. Also shown in Fig. 1 is a CPU 42 which communicates through a host bridge 44 with a PCI primary bus 46. Bridge 12 is also capable of communicating with the primary bus 46.

Please replace the paragraph beginning at page 3, line 25 with the following amended paragraph:

When an agent on the bus 30 requests a memory read operation, it notifies bridge 12 of the request by asserting the appropriate signals on the bus 30. If the bridge 12 determines that the request is for data from pre-fetchable storage 40, it multiplies a pre-defined amount of data requested by the number held in the pre-fetch factor register 15. The amount of data to be read depends upon the type of read request as well as the particular system design, for example the size of a cache line. Table 1 shows the data

amounts for three types of read requests. PFFR is the pre-fetch factor register value.

Please replace the table beginning at page 4, line 3 with the following amended table:

Memory Operation	Alignment	Read Size
Read	DWORD	(~~PPFR~~PFFR+1)*4*DWORD
Read Line	Cacheline	(~~PPFR~~PFFR +1)*cacheline
Read multiple	2 cachelines	(~~PPFR~~PFFR +1)*2 cachelines

Table 1

Please replace the paragraph beginning at page 4, line 13 with the following amended paragraph:

Referring to FIG 2, a flow chart of an adaptive read pre-fetch method 100 is shown. At system initialization 105, an initial value for the pre-fetch factor register is set. This may be in system ROM, or may be set (and changed from time to time) as a parameter by the operating system or any other system or application software. In one embodiment, a pre-fetch timer may be initialized to a set time, which will decrement to zero, unless reset.

Please replace the paragraph beginning at page 4, line 21 with the following amended paragraph:

If an agent gives a pre-fetchable read request 110 (of whatever type) then the read amount, based upon the type of read, (see table 1) is multiplied by the pre-fetch factor plus one at 120, the pre-fetch factor being stored –in the pre-fetch factor register 15–. Thus, if the value of the pre-fetch factor register is zero, the read amount is multiplied by one, effectively disabling the feature.

Please replace the paragraph beginning at page 5, line 3 with the following amended paragraph:

The value in the next read address register 28~~30~~ is compared to the value of the read address received from the agent at 125. If they are the same (meaning that the value in the next read address was stored as a result of a prior read request from the same agent which was terminated early for some reason, such as being disconnected by the bridge for lack of data), then the read amount is again increased. The read amount is multiplied by one plus the value in the re-read pre-fetch factor register 20 at 130. Other implementations could successively automatically increase the value in the re-read pre-fetch factor register 20 for each early terminated read, and conversely could periodically decrement the re-read pre-fetch factor.

Please replace the table beginning at page 5, line 21 with the following amended table:

Memory Operation	Alignment	Read Size
Read	DWORD	(~~PPFR~~PFFR +1+RRPFR)*4*DWORD
Read Line	cacheline	(~~PPFR~~PFFR +1+RRPFR)*cacheline
Read multiple	2 cachelines	(~~PPFR~~PFFR +1+RRPFR)*2cachelines

Please replace the paragraph beginning at page 5, line 23 with the following amended paragraph:

If the read terminates early, then the requesting agent has not received all of the data that ~~presumably~~ it presumably wants. Early termination occurs if the bridge 12 disconnects the read transaction because data is exhausted, and the requesting device is still expecting additional data (i.e. still asserting the PCI bus signal FRAME#.) Data may become exhausted because of a variety of reasons, including an end of file, exhaustion of a buffer or other causes.

Please replace the paragraph beginning at page 6, line 4 with the following amended paragraph:

In the case of a first early termination, the adaptive read pre-fetch process increases the amount of data retrieved on the next read request at the same location (where the current read ended) from the requesting agent. This is accomplished by saving the next-read address (the next address at which data would have been retrieved had the read not been terminated early) in the Next Read Address Register 28 at 145

and beginning to use the re-read pre-fetch factor and command type specific pre-fetch amounts.